Mint Classics Database Analysis and Optimization

Mahmoud Alsaleh

June 2024

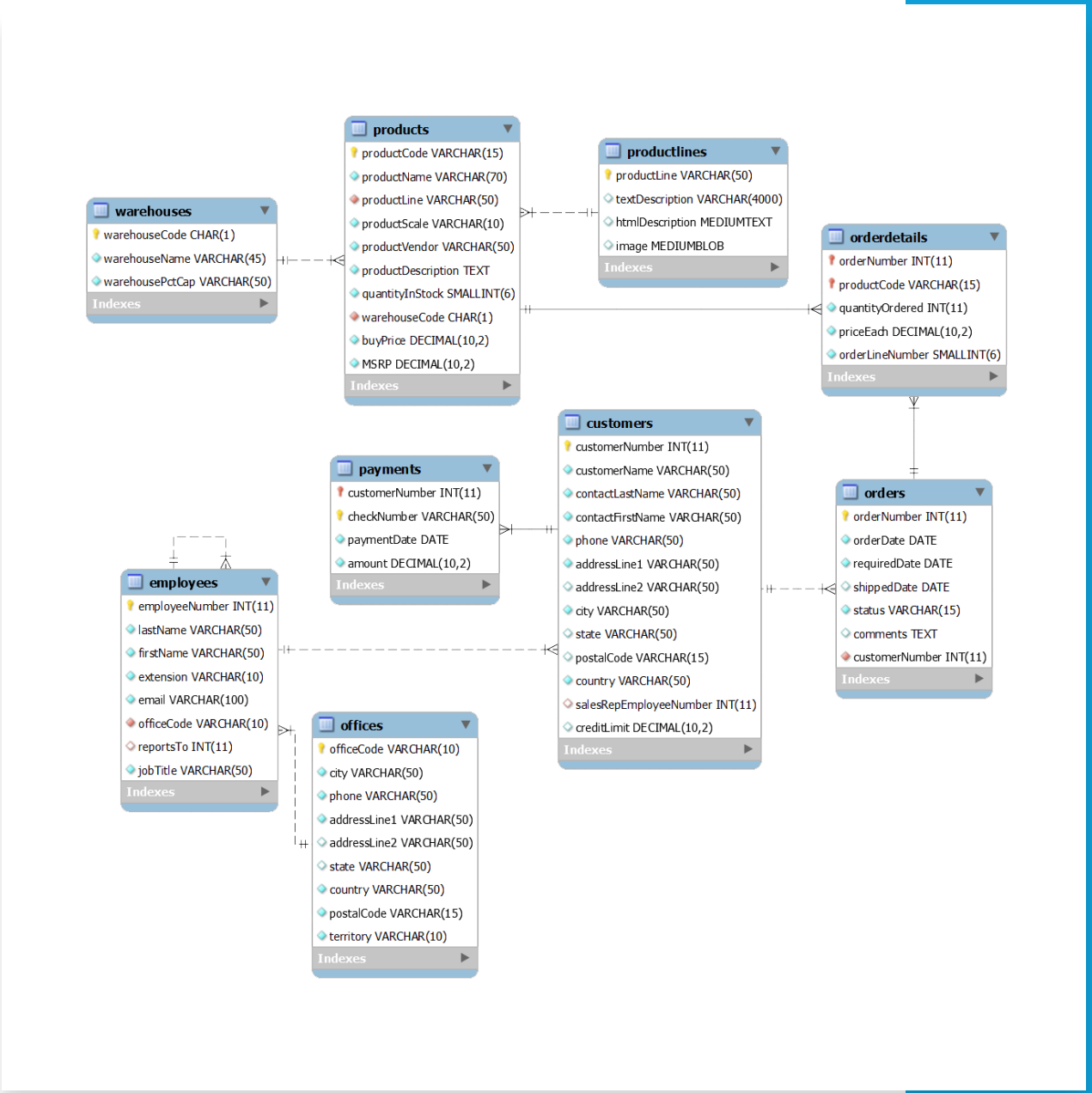
Introduction

This project aims to analyze and optimize the inventory management of Mint Classics Company by investigating the potential closure of one of its storage facilities. The project involves importing, understanding, and analyzing the Mint Classics relational database using MySQL Workbench. The analysis will lead to data-driven recommendations to improve business processes.

Data Overview

This project focuses on analyzing the Mint Classics Company's relational database, which includes comprehensive data on product inventory, sales, and storage locations. The dataset was provided by the company to assist in making data-driven decisions about inventory management, to close one of their storage facilities. By delving into this data, we aimed to uncover insights that would guide the reorganization or reduction of inventory while maintaining efficient customer service. Our analysis targeted key aspects such as inventory distribution, sales performance, and product movement to form a solid foundation for our recommendations.

Database Diagram



productCode	productName	productLine	productScale	productVendor	quantityInStock	warehouseCode	buyPrice	MSRP
S24_2011	18th century schooner	Ships	1:24	Carousel DieCast Legends	1898	d	82.34	122.89
S18_3136	18th Century Vintage Horse Carriage	Vintage Cars	1:18	Red Start Diecast	5992	c	60.74	104.72
S24_2841	1900s Vintage Bi-Plane	Planes	1:24	Autoart Studio Design	5942	a	34.25	68.51
S24_4278	1900s Vintage Tri-Plane	Planes	1:24	Unimax Art Galleries	2756	a	36.23	72.45
S18_3140	1903 Ford Model A	Vintage Cars	1:18	Unimax Art Galleries	3913	c	68.3	136.59
S18_4522	1904 Buick Runabout	Vintage Cars	1:18	Exoto Designs	8290	c	52.66	87.77
S18_2248	1911 Ford Town Car	Vintage Cars	1:18	Motor City Art Classics	540	c	33.3	60.54
S24_3151	1912 Ford Model T Delivery Wagon	Vintage Cars	1:24	Min Lin Diecast	9173	c	46.91	88.51
S18_2949	1913 Ford Model T Speedster	Vintage Cars	1:18	Carousel DieCast Legends	4189	c	60.78	101.31
S18_1749	1917 Grand Touring Sedan	Vintage Cars	1:18	Welly Diecast Productions	2724	c	86.7	170

Products

The Products table contains details about each product available in the Mint Classics Company's inventory, including product codes, names, scales, and prices. This information is crucial for identifying the variety of items stored and sold by the company.

Product Lines

The ProductLine table categorizes products into different lines or themes, providing a structured way to group related items. This helps in understanding the breadth and focus of the company's product offerings.

productLine
Classic Cars
Motorcycles
Planes
Ships
Trains
Trucks and Buses
Vintage Cars

orderNumber	orderDate	requiredDate	shippedDate	status	customerNumber
10100	06-01-03	13-01-03	10-01-03	Shipped	363
10101	09-01-03	18-01-03	11-01-03	Shipped	128
10102	10-01-03	18-01-03	14-01-03	Shipped	181
10103	29-01-03	07-02-03	02-02-03	Shipped	121
10104	31-01-03	09-02-03	01-02-03	Shipped	141
10105	11-02-03	21-02-03	12-02-03	Shipped	145
10106	17-02-03	24-02-03	21-02-03	Shipped	278
10107	24-02-03	03-03-03	26-02-03	Shipped	131
10108	03-03-03	12-03-03	08-03-03	Shipped	385
10109	10-03-03	19-03-03	11-03-03	Shipped	486

Orders

The Orders table records each customer order, including the order number, order date, and status. This data is essential for tracking sales and understanding customer demand over time.

Order Details

The OrderDetails table breaks down each order into specific items purchased, including quantities and prices. This granularity allows for a detailed analysis of sales and inventory movement at the product level.

orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber
10100	S18_1749	30	136	3
10100	S18_2248	50	55.09	2
10100	S18_4409	22	75.46	4
10100	S24_3969	49	35.29	1
10101	S18_2325	25	108.06	4
10101	S18_2795	26	167.06	1
10101	S24_1937	45	32.53	3
10101	S24_2022	46	44.35	2

Customer Table

The Customers table holds information about the company's customers, including contact details and credit limits. This table is key to understanding the customer base and their purchasing behaviors.

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	city	state	postalCode	country	salesRepEmployeeNumber	creditLimit
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale	Nantes	NULL	44000	France	1370	21000
112	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.	Las Vegas	NV	83030	USA	1166	71800
114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road	Melbourne	Victoria	3004	Australia	1611	117300
119	La Rochelle Gifts	Labrune	Janine	40.67.8555	67, rue des Cinquante Otages	Nantes	NULL	44000	France	1370	118200
121	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78	Stavern	NULL	4110	Norway	1504	81700
124	Mini Gifts Distributors Ltd.	Nelson	Susan	4155551450	5677 Strong St.	San Rafael	CA	97562	USA	1165	210500
125	Havel & Zbyszek Co	Piestrzeniewicz	Zbyszek	(26) 642-7555	ul. Filtrowa 68	Warszawa	NULL	01-012	Poland	NULL	0
128	Blauer See Auto, Co.	Keitel	Roland	+49 69 66 90 2555	Lyonerstr. 34	Frankfurt	NULL	60528	Germany	1504	59700
129	Mini Wheels Co.	Murphy	Julie	6505555787	5557 North Pendale Street	San Francisco	CA	94217	USA	1165	64600
131	Land of Toys Inc.	Lee	Kwai	2125557818	897 Long Airport Avenue	NYC	NY	10022	USA	1323	114900

Payments

customerNumber	checkNumber	paymentDate	amount
103	HQ336336	19-10-04	6066.78
103	JM555205	05-06-03	14571.44
103	OM314933	18-12-04	1676.14
112	BO864823	17-12-04	14191.12
112	HQ55022	06-06-03	32641.98
112	ND748579	20-08-04	33347.88
114	GG31455	20-05-03	45864.03
114	MA765515	15-12-04	82261.22
114	NP603840	31-05-03	7565.08
114	NR27552	10-03-04	44894.74

The Payments table documents payment transactions made by customers, including payment dates and amounts. This data is important for financial analysis and ensuring timely payment processing.

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (APAC)
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep

Employees

The Employees table lists information about the company's employees, including job titles, office locations, and supervisors. This table helps in analyzing the human resources aspect of the business.

officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	NULL	NULL	France	75017	EMEA
5	Tokyo	+81 33 224 5000	4-1 Kioicho	NULL	Chiyoda-Ku	Japan	102-8578	Japan
6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	Australia	NSW 2010	APAC
7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA

Offices

The Offices table provides details about the company’s office locations, including addresses and contact information. This data supports logistical and operational planning.

warehouseCode	warehouseName	warehousePctCap
a	North	72
b	East	67
c	West	50
d	South	75

Warehouses

The Warehouses table details the storage facilities, including warehouse names and capacities as percentages. This table is crucial for analyzing inventory distribution across different facilities and identifying opportunities for consolidation or reduction.

Database Explore

Unique Products in Each Product Line

productLine	Unique products
Classic Cars	38
Motorcycles	13
Planes	12
Ships	9
Trains	3
Trucks and Buses	11
Vintage Cars	24

```
Select productLine,
count(*) AS 'Unique products’
From mintclassics.products
group by productLine;
```

Total Capacity for each warehouse

```
SELECT p.warehouseCode, SUM(p.quantityInStock) AS total_in_warehouse,
w.warehousePctCap AS warehouse_Presentage_capacity,
ROUND((SUM(p.quantityInStock) / w.warehousePctCap) * 100, 0) AS Total_Capacity
FROM mintclassics.products p
join mintclassics.warehouses w
ON p.warehouseCode = w.warehouseCode
GROUP BY warehouseCode
```

warehouseCode	total_in_warehouse	warehouse_Presentage_capacity	Total_Capacity	empty_space
a	131688	72	182900	51212
b	219183	67	327139	107956
c	124880	50	249760	124880
d	79380	75	105840	26460

Production lines distribution in warehouses

```
SELECT warehouseCode,
productline
FROM mintclassics.products
Group by productline
ORDER BY warehouseCode
```

warehouseCode	productline
a	Motorcycles
a	Planes
b	Classic Cars
c	Vintage Cars
d	Ships
d	Trains
d	Trucks and Buses

Motorcycle Products

```
SELECT warehouseCode,
productCode, quantityInStock,
productline

FROM mintclassics.products

WHERE warehouseCode = "a" AND
productLine = 'Motorcycles’

ORDER BY quantityInStock;
```

* Warehouse (a) has 13 unique Motorcycle items and a total of 69401 Motorcycles.

warehouseCode	productCode	quantityInStock	productline	productScale
a	S24_2000	15	Motorcycles	1:24
a	S32_1374	178	Motorcycles	1:32
a	S50_4713	600	Motorcycles	1:50
a	S32_4485	3341	Motorcycles	1:32
a	S18_2625	4357	Motorcycles	1:18
a	S10_4698	5582	Motorcycles	1:10
a	S10_2016	6625	Motorcycles	1:10
a	S24_2360	6840	Motorcycles	1:24
a	S24_1578	7003	Motorcycles	1:24
a	S18_3782	7689	Motorcycles	1:18
a	S10_1678	7933	Motorcycles	1:10
a	S32_2206	9241	Motorcycles	1:32
a	S12_2823	9997	Motorcycles	1:12

Plane Products

warehouseCode	productCode	quantityInStock	productline	productScale
a	S700_3167	551	Planes	1:72
a	S18_2581	992	Planes	1:72
a	S24_4278	2756	Planes	1:24
a	S24_1785	3627	Planes	1:24
a	S72_1253	4857	Planes	1:72
a	S18_1662	5330	Planes	1:18
a	S700_1691	5841	Planes	1:700
a	S24_2841	5942	Planes	1:24
a	S24_3949	6812	Planes	1:24
a	S700_2834	7106	Planes	1:700
a	S700_4002	8820	Planes	1:700
a	S700_2466	9653	Planes	1:700

```
SELECT warehouseCode, productCode,
quantityInStock, productline

FROM mintclassics.products

WHERE warehouseCode = "a" AND
productLine = Planes’

ORDER BY quantityInStock;
```

* Warehouse (a) has 12 unique Plane items and a total of 62287 Planes.

Classic Car Products (1/2)

```
SELECT warehouseCode, productCode, quantityInStock, productline
FROM mintclassics.products
WHERE warehouseCode = “b”
ORDER BY quantityInStock;
```

* Warehouse (b) has only Classic Cars and it has 38 unique car items and a total of 219183 cars.

warehouseCode	productCode	quantityInStock	productScale	productline
b	S12_1099	68	1:12	Classic Cars
b	S24_1046	1005	1:24	Classic Cars
b	S12_3891	1049	1:12	Classic Cars
b	S18_4721	1249	1:18	Classic Cars
b	S24_2887	1452	1:24	Classic Cars
b	S18_3278	1917	1:18	Classic Cars
b	S24_2766	2350	1:24	Classic Cars
b	S24_2840	2542	1:24	Classic Cars
b	S18_4933	3209	1:18	Classic Cars
b	S10_4757	3252	1:10	Classic Cars
b	S12_1108	3619	1:12	Classic Cars
b	S18_1129	3975	1:18	Classic Cars
b	S24_1444	4074	1:24	Classic Cars
b	S24_3191	4695	1:24	Classic Cars
b	S18_2238	4724	1:18	Classic Cars
b	S18_4027	5545	1:18	Classic Cars
b	S12_3990	5663	1:12	Classic Cars
b	S24_4048	6582	1:24	Classic Cars
b	S24_3856	6600	1:18	Classic Cars

Classic Car Products (2/2)

```
SELECT warehouseCode,
productCode, quantityInStock,
productline
FROM mintclassics.products
WHERE warehouseCode = “b”
ORDER BY quantityInStock;
```

* Warehouse (b) has only Classic Cars and it has 38 unique car items and a total of 219183 cars.

warehouseCode	productCode	quantityInStock	productScale	productline
b	S10_4962	6791	1:10	Classic Cars
b	S12_3148	6906	1:18	Classic Cars
b	S700_2824	6934	1:18	Classic Cars
b	S10_1949	7305	1:10	Classic Cars
b	S12_4675	7323	1:12	Classic Cars
b	S24_2972	7723	1:24	Classic Cars
b	S18_3233	7733	1:18	Classic Cars
b	S24_4620	7869	1:18	Classic Cars
b	S24_3371	7995	1:24	Classic Cars
b	S18_2870	8164	1:18	Classic Cars
b	S24_1628	8197	1:24	Classic Cars
b	S18_3232	8347	1:18	Classic Cars
b	S18_1889	8826	1:18	Classic Cars
b	S18_3685	8990	1:18	Classic Cars
b	S18_1589	9042	1:18	Classic Cars
b	S12_3380	9123	1:12	Classic Cars
b	S18_3482	9127	1:18	Classic Cars
b	S24_3432	9446	1:24	Classic Cars
b	S18_1984	9772	1:18	Classic Cars

Vintage Car Products

```
SELECT warehouseCode,
productCode, quantityInStock,
productline
FROM mintclassics.products
WHERE warehouseCode = “c”
ORDER BY quantityInStock;
```

* Warehouse (c) has only Vintage Cars and it has 24 unique car items and a total of 124880 cars.

warehouseCode	productCode	quantityInStock	productScale	productline
c	S32_4289	136	1:32	Vintage Cars
c	S18_2248	540	1:18	Vintage Cars
c	S18_2795	548	1:18	Vintage Cars
c	S24_3969	2081	1:24	Vintage Cars
c	S18_3856	2378	1:18	Vintage Cars
c	S18_1749	2724	1:18	Vintage Cars
c	S24_2022	2847	1:24	Vintage Cars
c	S24_3420	2902	1:24	Vintage Cars
c	S18_3140	3913	1:18	Vintage Cars
c	S18_2949	4189	1:18	Vintage Cars
c	S24_4258	4710	1:24	Vintage Cars
c	S18_2957	5649	1:18	Vintage Cars
c	S18_3136	5992	1:18	Vintage Cars
c	S18_4409	6553	1:18	Vintage Cars
c	S24_3816	6621	1:24	Vintage Cars
c	S18_4668	6645	1:18	Vintage Cars
c	S50_1341	7062	1:50	Vintage Cars
c	S24_1937	7332	1:24	Vintage Cars
c	S18_3320	7913	1:18	Vintage Cars
c	S18_4522	8290	1:18	Vintage Cars
c	S18_1367	8635	1:18	Vintage Cars
c	S18_1342	8693	1:18	Vintage Cars
c	S24_3151	9173	1:24	Vintage Cars
c	S18_2325	9354	1:18	Vintage Cars

Ship Products

```
SELECT warehouseCode,
productCode, quantityInStock,
productline

FROM mintclassics.products

WHERE warehouseCode = “d” AND
productline = 'Ships'

ORDER BY quantityInStock;
```

* Warehouse (d) has 9 unique Ship items and a total of 26833 Ships.

warehouseCode	productCode	quantityInStock	productline	productScale
d	S72_3212	414	Ships	1:27
d	S700_1938	737	Ships	1:700
d	S700_1138	1897	Ships	1:700
d	S24_2011	1898	Ships	1:24
d	S700_3505	1956	Ships	1:700
d	S700_2047	3501	Ships	1:700
d	S18_3029	4259	Ships	1:18
d	S700_3962	5088	Ships	1:700
d	S700_2610	7083	Ships	1:700

Train Products

```
SELECT warehouseCode, productCode, quantityInStock,
productline
FROM mintclassics.products
WHERE warehouseCode = “d” AND productline = ‘Trains'
ORDER BY quantityInStock;
```

warehouseCode	productCode	quantityInStock	productline
d	S50_1514	1645	Trains
d	S18_3259	6450	Trains
d	S32_3207	8601	Trains

* Warehouse (d) has 3 unique Train items and a total of 16696 Trains.

Truck & Bus Products

```
SELECT warehouseCode, productCode, quantityInStock, productline
FROM mintclassics.products
WHERE warehouseCode = “d” AND productline = ‘Truck and Buses'
ORDER BY quantityInStock;
```

warehouseCode	productCode	quantityInStock	productline	productScale
d	S32_3522	814	Trucks and Buses	1:32
d	S50_1392	1016	Trucks and Buses	1:50
d	S12_1666	1579	Trucks and Buses	1:12
d	S18_2432	2018	Trucks and Buses	1:18
d	S24_2300	2327	Trucks and Buses	1:24
d	S18_1097	2613	Trucks and Buses	1:18
d	S32_2509	2874	Trucks and Buses	1:32
d	S18_4600	3128	Trucks and Buses	1:18
d	S32_1268	5099	Trucks and Buses	1:32
d	S12_4473	6125	Trucks and Buses	1:12
d	S18_2319	8258	Trucks and Buses	1:18

* Warehouse (d) has 11 unique Truck and bus items and a total of 35851 Trucks and Buses.

Revenue

```
SELECT sub.productCode, p.productName, sub.avgSellPrice AS sellPrice, p.buyPrice, sub.totalSold, (sub.avgSellPrice - p.buyPrice) * sub.totalSold AS revenue
FROM(
               SELECT od.productCode,
               AVG(od.priceEach) AS avgSellPrice,
             SUM(od.quantityOrdered) AS totalSold
               FROM mintclassics.orderdetails od
               JOIN mintclassics.orders o
               ON od.orderNumber = o.orderNumber
               WHERE o.status IN ('Shipped', 'Resolved’)
               GROUP BY od.productCode
     ) AS sub
JOIN mintclassics.products p ON sub.productCode = p.productCode
ORDER BY revenue DESC;
```

productCode	productName	sellPrice	buyPrice	totalSold	revenue
S18_3232	1992 Ferrari 360 Spider red	152.8132	77.9	1720	128850.704
S10_1949	1952 Alpine Renault 1300	197.1563	98.58	911	89803.0057
S12_1108	2001 Ferrari Enzo	187.33923	95.59	973	89272.0018
S10_4698	2003 Harley-Davidson Eagle Drag Bike	172.64444	91.02	929	75829.1085
S12_1099	1968 Ford Mustang	173.01731	95.34	909	70608.673
S12_3891	1969 Ford Falcon	157.78269	83.05	921	68828.8093
S12_2823	2002 Suzuki XREO	131.70852	66.27	1007	65896.5886
S18_2795	1928 Mercedes-Benz SSK	149.25148	72.56	845	64804.3014
S18_4721	1957 Corvette Convertible	129.29074	69.93	1013	60132.4306
S18_1662	1980s Black Hawk Helicopter	138.28231	77.27	948	57839.668
S18_3685	1948 Porsche Type 356 Roadster	129.11783	62.16	852	57048.0678
S18_1749	1917 Grand Touring Sedan	154.25652	86.7	805	54383.0002
S18_3482	1976 Ford Gran Torino	132.50778	73.49	915	54001.2669
S18_2325	1932 Model A Ford J-Coupe	114.13519	58.48	912	50757.5287
S24_2300	1962 Volkswagen Microbus	114.71577	61.34	938	50066.4713
S12_4473	1957 Chevy Pickup	104.19519	55.7	1023	49610.5743
S18_2870	1999 Indy 500 Monte Carlo SS	117.93913	56.76	777	47536.184
S12_3148	1969 Corvair Monza	137.65846	89.14	933	45267.725
S18_1097	1940 Ford Pickup Truck	105.39222	58.33	945	44473.7998
S18_3140	1903 Ford Model A	125.43583	68.3	778	44451.6781

Top 20

Monthly Average Revenue

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, od.priceEach,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)
SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item, ts.total_revenue,
ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- Assuming 18-month period for average sales calculation
ROUND(ts.priceEach * (ts.total_ordered_item / 18)) AS avg_monthly_revenue -- Assuming 18-month period for average revenue calculation
FROM TotalSales ts
ORDER BY avg_monthly_revenue DESC
LIMIT 20
```

productCode	productName	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	avg_monthly_revenue
S18_3232	1992 Ferrari 360 Spider red	b	8347	1808	276839.98	100	16669
S12_1108	2001 Ferrari Enzo	b	3619	1019	190755.86	57	11646
S10_1949	1952 Alpine Renault 1300	b	7305	961	190017.96	53	11441
S10_4698	2003 Harley-Davidson Eagle Drag Bike	a	5582	985	170686	55	9432
S12_1099	1968 Ford Mustang	b	68	933	161531.48	52	8572
S18_2795	1928 Mercedes-Benz SSK	c	548	880	132275.98	49	8167
S18_4721	1957 Corvette Convertible	b	1249	1013	130749.31	56	7872
S18_1662	1980s Black Hawk Helicopter	a	5330	1040	144959.91	58	7745
S24_3856	1956 Porsche 356A Coupe	b	6600	1052	134240.71	58	7715
S12_3891	1969 Ford Falcon	b	1049	965	152543.02	54	7606
S18_2238	1998 Chrysler Plymouth Prowler	b	4724	986	142530.63	55	7444
S10_4757	1972 Alfa Romeo GTA	b	3252	1030	127924.32	57	7315
S18_2319	1964 Mercedes Tour Bus	d	8258	1053	117669.66	59	7180
S12_3148	1969 Corvair Monza	b	6906	963	132363.79	54	7032
S18_1984	1995 Honda Civic	b	9772	917	119050.95	51	7029
S12_2823	2002 Suzuki XREO	a	9997	1028	135767.03	57	6968
S18_1749	1917 Grand Touring Sedan	c	2724	918	140535.6	51	6936
S18_3482	1976 Ford Gran Torino	b	9127	915	121890.6	51	6725
S18_3140	1903 Ford Model A	c	3913	883	111528.82	49	6700
S24_2011	18th century schooner	d	1898	1011	112427.12	56	6626

Top 20

productCode	warehouseCode	quantityInStock	total_ordered_item
S18_3233	b	7733	0

Never ordered Items

```
SELECT p.productCode, p.warehouseCode, p.quantityInStock,
COALESCE(SUM(CASE WHEN o.status IN ('Shipped', 'Resolved’)
THEN od.quantityOrdered ELSE 0 END), 0) AS total_ordered_item
FROM mintclassics.products p
LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber
GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
HAVING total_ordered_item < 1
ORDER BY p.warehouseCode;
```

Recommendations and Action Plan

Good Stock level (Using Old sales)

```
WITH TotalSales AS (
      SELECT p.productCode, p.warehouseCode, p.quantityInStock,
      COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item
      FROM mintclassics.products p
      LEFT JOIN mintclassics.orderdetails od ON p.productCode =
      od.productCode
      LEFT JOIN mintclassics.orders o ON od.orderNumber =
      o.orderNumber AND o.status IN ('Shipped', 'Resolved’)
      GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)

SELECT ts.productCode, ts.warehouseCode,
ts.quantityInStock,ts.total_ordered_item,

ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- 18-
month period for average sales calculation

ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level --
Assuming 12 months buffer for good stock level

FROM TotalSales ts

ORDER BY ROUND((ts.total_ordered_item / 18) * 12, 0)
```

Top 10 and Last 10

productCode	warehouseCode	quantityInStock	total_ordered_item	avg_monthly_sales	good_stock_level
S18_3232	b	8347	1808	100	1205
S18_1342	c	8693	1111	62	741
S700_4002	a	8820	1085	60	723
S18_3856	c	2378	1076	60	717
S50_1341	c	7062	1074	60	716
S18_4600	d	3128	1061	59	707
S10_1678	a	7933	1057	59	705
S12_4473	d	6125	1056	59	704
S18_2319	d	8258	1053	59	702
S24_3856	b	6600	1052	58	701
S24_2887	b	1452	873	49	582
S24_3191	b	4695	870	48	580
S24_4048	b	6582	867	48	578
S18_4409	c	6553	866	48	577
S18_2870	b	8164	855	48	570
S18_2248	c	540	832	46	555
S24_3969	c	2081	824	46	549
S24_1046	b	1005	803	45	535
S18_4933	b	3209	767	43	511
S18_3233	b	7733	0	0	0

Overstocked A Warehouse

```
WITH TotalSales AS (
       SELECT p.productCode, p.warehouseCode, p.quantityInStock,
       COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item
       FROM mintclassics.products p
       LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
       LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved’)
       GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)
SELECT ts.productCode, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item,
ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- 18-month period for average sales calculation
ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level, -- Assuming 12 months buffer for good stock level
CASE
        WHEN ts.total_ordered_item < 1 OR ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked’
END AS inventory_status
FROM TotalSales ts
WHERE ts.quantityInStock > (ts.total_ordered_item / 18) * 12 AND  ts.warehouseCode = 'a’
ORDER BY ts.warehouseCode, good_stock_level
```

A Warehouse

productCode	warehouseCode	quantityInStock	total_ordered_item	avg_monthly_sales	good_stock_level	inventory_status
S700_1691	a	5841	894	50	596	Overstocked
S32_4485	a	3341	898	50	599	Overstocked
S32_2206	a	9241	906	50	604	Overstocked
S18_2581	a	992	917	51	611	Overstocked
S24_2841	a	5942	940	52	627	Overstocked
S18_2625	a	4357	945	53	630	Overstocked
S24_2360	a	6840	947	53	631	Overstocked
S18_3782	a	7689	959	53	639	Overstocked
S72_1253	a	4857	960	53	640	Overstocked
S24_1785	a	3627	972	54	648	Overstocked
S700_2834	a	7106	973	54	649	Overstocked
S700_2466	a	9653	984	55	656	Overstocked
S10_4698	a	5582	985	55	657	Overstocked
S10_2016	a	6625	999	56	666	Overstocked
S24_4278	a	2756	1009	56	673	Overstocked
S12_2823	a	9997	1028	57	685	Overstocked
S24_1578	a	7003	1033	57	689	Overstocked
S18_1662	a	5330	1040	58	693	Overstocked
S24_3949	a	6812	1051	58	701	Overstocked
S10_1678	a	7933	1057	59	705	Overstocked
S700_4002	a	8820	1085	60	723	Overstocked

Overstocked B Warehouse

```
WITH TotalSales AS (
        SELECT p.productCode, p.warehouseCode, p.quantityInStock,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved’)
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)
SELECT ts.productCode, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item,
ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- 18-month period for average sales calculation
ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level, -- Assuming 12 months buffer for good stock level
CASE
        WHEN ts.total_ordered_item < 1 OR ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked’
END AS inventory_status
FROM TotalSales ts
WHERE ts.quantityInStock > (ts.total_ordered_item / 18) * 12 AND  ts.warehouseCode = ‘b’
ORDER BY ts.warehouseCode, good_stock_level
```

B Warehouse

productCode	warehouseCode	quantityInStock	total_ordered_item	avg_monthly_sales	good_stock_level	inventory_status
S18_3233	b	7733	0	0	0	Overstocked
S18_4933	b	3209	767	43	511	Overstocked
S24_1046	b	1005	803	45	535	Overstocked
S18_2870	b	8164	855	48	570	Overstocked
S24_4048	b	6582	867	48	578	Overstocked
S24_3191	b	4695	870	48	580	Overstocked
S24_2887	b	1452	873	49	582	Overstocked
S24_3432	b	9446	894	50	596	Overstocked
S12_3990	b	5663	900	50	600	Overstocked
S24_2972	b	7723	912	51	608	Overstocked
S18_1589	b	9042	914	51	609	Overstocked
S24_1628	b	8197	915	51	610	Overstocked
S18_3482	b	9127	915	51	610	Overstocked
S18_1984	b	9772	917	51	611	Overstocked
S12_3380	b	9123	925	51	617	Overstocked
S10_4962	b	6791	932	52	621	Overstocked
S24_4620	b	7869	941	52	627	Overstocked
S18_4027	b	5545	945	53	630	Overstocked
S18_1129	b	3975	947	53	631	Overstocked
S18_3685	b	8990	948	53	632	Overstocked
S24_2766	b	2350	949	53	633	Overstocked
S10_1949	b	7305	961	53	641	Overstocked
S12_3148	b	6906	963	54	642	Overstocked
S12_3891	b	1049	965	54	643	Overstocked
S24_3371	b	7995	969	54	646	Overstocked
S18_1889	b	8826	972	54	648	Overstocked
S18_3278	b	1917	974	54	649	Overstocked
S24_1444	b	4074	976	54	651	Overstocked
S24_2840	b	2542	983	55	655	Overstocked
S18_2238	b	4724	986	55	657	Overstocked
S12_4675	b	7323	992	55	661	Overstocked
S700_2824	b	6934	997	55	665	Overstocked
S18_4721	b	1249	1013	56	675	Overstocked
S12_1108	b	3619	1019	57	679	Overstocked
S10_4757	b	3252	1030	57	687	Overstocked
S24_3856	b	6600	1052	58	701	Overstocked
S18_3232	b	8347	1808	100	1205	Overstocked

Overstocked C Warehouse

```
WITH TotalSales AS (
       SELECT p.productCode, p.warehouseCode, p.quantityInStock,
       COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item
       FROM mintclassics.products p
       LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
       LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved’)
       GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)
SELECT ts.productCode, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item,
ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- 18-month period for average sales calculation
ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level, -- Assuming 12 months buffer for good stock level
CASE
        WHEN ts.total_ordered_item < 1 OR ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN
'Overstocked’
END AS inventory_status
FROM TotalSales ts
WHERE ts.quantityInStock > (ts.total_ordered_item / 18) * 12 AND  ts.warehouseCode = ‘c’
ORDER BY ts.warehouseCode, good_stock_level
```

C Warehouse

productCode	warehouseCode	quantityInStock	total_ordered_item	avg_monthly_sales	good_stock_level	inventory_status
S24_3969	c	2081	824	46	549	Overstocked
S18_4409	c	6553	866	48	577	Overstocked
S18_3140	c	3913	883	49	589	Overstocked
S24_3420	c	2902	884	49	589	Overstocked
S18_3136	c	5992	907	50	605	Overstocked
S18_1749	c	2724	918	51	612	Overstocked
S24_3816	c	6621	923	51	615	Overstocked
S24_1937	c	7332	937	52	625	Overstocked
S24_2022	c	2847	955	53	637	Overstocked
S18_2325	c	9354	957	53	638	Overstocked
S18_1367	c	8635	960	53	640	Overstocked
S24_4258	c	4710	983	55	655	Overstocked
S18_2957	c	5649	985	55	657	Overstocked
S18_4522	c	8290	990	55	660	Overstocked
S18_3320	c	7913	992	55	661	Overstocked
S24_3151	c	9173	991	55	661	Overstocked
S18_4668	c	6645	995	55	663	Overstocked
S18_2949	c	4189	1038	58	692	Overstocked
S50_1341	c	7062	1074	60	716	Overstocked
S18_3856	c	2378	1076	60	717	Overstocked
S18_1342	c	8693	1111	62	741	Overstocked

Overstocked D Warehouse

```
WITH TotalSales AS (
        SELECT p.productCode, p.warehouseCode, p.quantityInStock,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved’)
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)
SELECT ts.productCode, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item,
ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- 18-month period for average sales calculation
ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level, -- Assuming 12 months buffer for good stock level
CASE
         WHEN ts.total_ordered_item < 1 OR ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked’
END AS inventory_status
FROM TotalSales ts
WHERE ts.quantityInStock > (ts.total_ordered_item / 18) * 12 AND  ts.warehouseCode = ‘d’
ORDER BY ts.warehouseCode, good_stock_level
```

productCode	warehouseCode	quantityInStock	total_ordered_item	avg_monthly_sales	good_stock_level	inventory_status
S700_3962	d	5088	896	50	597	Overstocked
S700_2047	d	3501	897	50	598	Overstocked
S700_1938	d	737	898	50	599	Overstocked
S32_1268	d	5099	911	51	607	Overstocked
S18_3259	d	6450	918	51	612	Overstocked
S700_1138	d	1897	934	52	623	Overstocked
S32_3207	d	8601	934	52	623	Overstocked
S700_3505	d	1956	952	53	635	Overstocked
S32_2509	d	2874	955	53	637	Overstocked
S50_1514	d	1645	966	54	644	Overstocked
S18_3029	d	4259	966	54	644	Overstocked
S12_1666	d	1579	972	54	648	Overstocked
S50_1392	d	1016	979	54	653	Overstocked
S32_3522	d	814	988	55	659	Overstocked
S18_2432	d	2018	998	55	665	Overstocked
S18_1097	d	2613	999	56	666	Overstocked
S24_2011	d	1898	1011	56	674	Overstocked
S700_2610	d	7083	1020	57	680	Overstocked
S24_2300	d	2327	1029	57	686	Overstocked
S18_2319	d	8258	1053	59	702	Overstocked
S12_4473	d	6125	1056	59	704	Overstocked
S18_4600	d	3128	1061	59	707	Overstocked

D Warehouse

Overstocked Items Across Warehouses

All four warehouses have overstocked items, with quantities exceeding optimal stock levels. This overstocking leads to increased storage costs and potential losses. Addressing these issues is crucial for optimizing inventory management and warehouse efficiency. By reallocating or liquidating these items, we can reduce costs and improve operational efficiency across all locations.

Warehouse	Overstocked Items
A	21
B	15
C	18
D	10

Understocked

```
WITH TotalSales AS (
        SELECT p.productCode, p.warehouseCode, p.quantityInStock,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved’)
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock
)
SELECT ts.productCode, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item,
ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales, -- 18-month period for average sales calculation
ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level, -- Assuming 12 months buffer for good stock level
CASE
        WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN Understocked’
END AS inventory_status
FROM TotalSales ts
WHERE ts.quantityInStock < (ts.total_ordered_item / 18) * 12
ORDER BY ts.warehouseCode, good_stock_level
```

productCode	warehouseCode	quantityInStock	total_ordered_item	avg_monthly_sales	good_stock_level	inventory_status
S50_4713	a	600	992	55	661	Understocked
S32_1374	a	178	1014	56	676	Understocked
S24_2000	a	15	1015	56	677	Understocked
S700_3167	a	551	1047	58	698	Understocked
S12_1099	b	68	933	52	622	Understocked
S18_2248	c	540	832	46	555	Understocked
S18_2795	c	548	880	49	587	Understocked
S32_4289	c	136	972	54	648	Understocked
S72_3212	d	414	958	53	639	Understocked

Understocked

Understocked Items Across Warehouses

Warehouse	Understocked Items
A	4
B	1
C	3
D	1

All four warehouses have understocked items, with quantities below optimal stock levels. Specifically, Warehouse A has 4 understocked items, Warehouse B has 1, Warehouse C has 3, and Warehouse D has 1. Addressing these understocked items is crucial for preventing stockouts and ensuring customer satisfaction. By replenishing these items, we can maintain adequate inventory levels and improve operational efficiency.

Actions

Reduce in A Warehouse

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, p.productScale,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped',
        'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock, p.productScale

),

InventoryAnalysis AS (
        SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock,
        ts.total_ordered_item, ts.total_revenue,
        ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales,
        ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level_sales,
        ROUND(ts.total_revenue / 18, 0) AS avg_monthly_revenue, ts.productScale,
        CASE
                WHEN ts.total_ordered_item < 1 THEN 'Never ordered'
                WHEN ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked'
                WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN 'Understocked'
                ELSE 'Well-Stocked'
        END AS inventory_status
        FROM TotalSales ts

)
```

```
SELECT ia.productCode, ia.warehouseCode, ia.quantityInStock, ia.total_ordered_item,
ia.total_revenue,

ia.avg_monthly_sales, ia.good_stock_level_sales, ia.avg_monthly_revenue,
ia.inventory_status,

CASE
        WHEN ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked'
        AND
        (ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR
        ia.productScale LIKE '%1:50%'
        OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%') THEN 'Reduce'
        WHEN ia.inventory_status = 'Understocked' THEN 'Increase'
        ELSE 'Maintain'

END AS action

FROM InventoryAnalysis ia

where ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND

(ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE
'%1:50%'

OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%')

HAVING ia.warehouseCode = "a"

ORDER BY ia.warehouseCode
```

productCode	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	good_stock_level_sales	avg_monthly_revenue	inventory_status	action
S24_3949	a	6812	1051	62269.67	58	701	3459	Overstocked	Reduce
S18_2581	a	992	917	68741.91	51	611	3819	Overstocked	Reduce
S700_2466	a	9653	984	89347.8	55	656	4964	Overstocked	Reduce
S24_2360	a	6840	947	57995.25	53	631	3222	Overstocked	Reduce
S24_4278	a	2756	1009	68276.35	56	673	3793	Overstocked	Reduce
S700_2834	a	7106	973	102786.38	54	649	5710	Overstocked	Reduce
S24_1578	a	7003	1033	105266.64	57	689	5848	Overstocked	Reduce
S32_2206	a	9241	906	33268.76	50	604	1848	Overstocked	Reduce
S24_1785	a	3627	972	94885.37	54	648	5271	Overstocked	Reduce
S700_1691	a	5841	894	73871.22	50	596	4104	Overstocked	Reduce
S72_1253	a	4857	960	42692.53	53	640	2372	Overstocked	Reduce
S32_4485	a	3341	898	84039.24	50	599	4669	Overstocked	Reduce
S24_2841	a	5942	940	58434.07	52	627	3246	Overstocked	Reduce
S700_4002	a	8820	1085	71753.93	60	723	3986	Overstocked	Reduce

A Warehouse

Reduce in B Warehouse

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, p.productScale,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped',
        'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock, p.productScale

),

InventoryAnalysis AS (
        SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock,
        ts.total_ordered_item, ts.total_revenue,
        ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales,
        ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level_sales,
        ROUND(ts.total_revenue / 18, 0) AS avg_monthly_revenue, ts.productScale,
        CASE
                WHEN ts.total_ordered_item < 1 THEN 'Never ordered'
                WHEN ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked'
                WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN 'Understocked'
                ELSE 'Well-Stocked'
        END AS inventory_status
        FROM TotalSales ts

)
```

```
SELECT ia.productCode, ia.warehouseCode, ia.quantityInStock, ia.total_ordered_item, ia.total_revenue,

ia.avg_monthly_sales, ia.good_stock_level_sales, ia.avg_monthly_revenue, ia.inventory_status,

CASE
        WHEN ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND
        (ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE
        '%1:50%'
        OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%') THEN 'Reduce'
        WHEN ia.inventory_status = 'Understocked' THEN 'Increase'
        ELSE 'Maintain'

END AS action

FROM InventoryAnalysis ia

where ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND

(ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE '%1:50%'

OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%')

HAVING ia.warehouseCode = “b"

ORDER BY ia.warehouseCode
```

B Warehouse

productCode	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	good_stock_level_sales	avg_monthly_revenue	inventory_status	action
S24_2972	b	7723	912	30972.87	51	608	1721	Overstocked	Reduce
S24_4048	b	6582	867	92973.4	48	578	5165	Overstocked	Reduce
S24_1046	b	1005	803	53236.67	45	535	2958	Overstocked	Reduce
S24_3191	b	4695	870	67357.3	48	580	3742	Overstocked	Reduce
S24_2840	b	2542	983	31627.96	55	655	1757	Overstocked	Reduce
S24_2887	b	1452	873	94248.67	49	582	5236	Overstocked	Reduce
S18_3233	b	7733	0	0	0	0	0	Never ordered	Reduce
S24_1444	b	4074	976	50255.45	54	651	2792	Overstocked	Reduce
S24_3371	b	7995	969	52339.53	54	646	2908	Overstocked	Reduce
S24_2766	b	2350	949	76670.02	53	633	4259	Overstocked	Reduce
S24_1628	b	8197	915	42015.54	51	610	2334	Overstocked	Reduce
S24_3432	b	9446	894	87404.81	50	596	4856	Overstocked	Reduce

Reduce in C Warehouse

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, p.productScale,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped',
        'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock, p.productScale

),

InventoryAnalysis AS (
        SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock,
        ts.total_ordered_item, ts.total_revenue,
        ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales,
        ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level_sales,
        ROUND(ts.total_revenue / 18, 0) AS avg_monthly_revenue, ts.productScale,
        CASE
                WHEN ts.total_ordered_item < 1 THEN 'Never ordered'
                WHEN ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked'
                WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN 'Understocked'
                ELSE 'Well-Stocked'
        END AS inventory_status
        FROM TotalSales ts

)
```

```
SELECT ia.productCode, ia.warehouseCode, ia.quantityInStock, ia.total_ordered_item, ia.total_revenue,

ia.avg_monthly_sales, ia.good_stock_level_sales, ia.avg_monthly_revenue, ia.inventory_status,

CASE
        WHEN ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND
        (ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE
        '%1:50%'
        OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%') THEN 'Reduce'
        WHEN ia.inventory_status = 'Understocked' THEN 'Increase'
        ELSE 'Maintain'

END AS action

FROM InventoryAnalysis ia

where ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND

(ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE '%1:50%'

OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%')

HAVING ia.warehouseCode = “c"

ORDER BY ia.warehouseCode
```

C Warehouse

productCode	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	good_stock_level_sales	avg_monthly_revenue	inventory_status	action
S24_1937	c	7332	937	28052.94	52	625	1558	Overstocked	Reduce
S24_2022	c	2847	955	38449.09	53	637	2136	Overstocked	Reduce
S50_1341	c	7062	1074	41599.24	60	716	2311	Overstocked	Reduce
S24_3420	c	2902	884	52803.75	49	589	2934	Overstocked	Reduce
S24_3816	c	6621	923	71208.18	51	615	3956	Overstocked	Reduce
S24_3969	c	2081	824	29763.39	46	549	1654	Overstocked	Reduce
S24_3151	c	9173	991	77239.92	55	661	4291	Overstocked	Reduce
S24_4258	c	4710	983	88434.46	55	655	4913	Overstocked	Reduce

Reduce in D Warehouse

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, p.productScale,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped',
        'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock, p.productScale

),

InventoryAnalysis AS (
        SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock,
        ts.total_ordered_item, ts.total_revenue,
        ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales,
        ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level_sales,
        ROUND(ts.total_revenue / 18, 0) AS avg_monthly_revenue, ts.productScale,
        CASE
                WHEN ts.total_ordered_item < 1 THEN 'Never ordered'
                WHEN ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked'
                WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN 'Understocked'
                ELSE 'Well-Stocked'
        END AS inventory_status
        FROM TotalSales ts

)
```

```
SELECT ia.productCode, ia.warehouseCode, ia.quantityInStock, ia.total_ordered_item, ia.total_revenue,

ia.avg_monthly_sales, ia.good_stock_level_sales, ia.avg_monthly_revenue, ia.inventory_status,

CASE
        WHEN ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND
        (ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE
        '%1:50%'
        OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%') THEN 'Reduce'
        WHEN ia.inventory_status = 'Understocked' THEN 'Increase'
        ELSE 'Maintain'

END AS action

FROM InventoryAnalysis ia

where ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND

(ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE '%1:50%'

OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%')

HAVING ia.warehouseCode = “d"

ORDER BY ia.warehouseCode
```

D Warehouse

productCode	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	good_stock_level_sales	avg_monthly_revenue	inventory_status	action
S700_1938	d	737	898	69531.61	50	599	3863	Overstocked	Reduce
S32_3522	d	814	988	57282.49	55	659	3182	Overstocked	Reduce
S50_1514	d	1645	966	52123.81	54	644	2896	Overstocked	Reduce
S700_3962	d	5088	896	78919.06	50	597	4384	Overstocked	Reduce
S32_3207	d	8601	934	53791.99	52	623	2988	Overstocked	Reduce
S24_2011	d	1898	1011	112427.12	56	674	6246	Overstocked	Reduce
S700_2047	d	3501	897	73298.42	50	598	4072	Overstocked	Reduce
S24_2300	d	2327	1029	118774.33	57	686	6599	Overstocked	Reduce
S700_2610	d	7083	1020	66697.13	57	680	3705	Overstocked	Reduce
S50_1392	d	1016	979	101137.55	54	653	5619	Overstocked	Reduce
S32_1268	d	5099	911	78067.82	51	607	4337	Overstocked	Reduce
S700_3505	d	1956	952	84992.25	53	635	4722	Overstocked	Reduce
S700_1138	d	1897	934	56455.11	52	623	3136	Overstocked	Reduce
S32_2509	d	2874	955	46519.05	53	637	2584	Overstocked	Reduce

Items to Reduce for Space Optimization

To optimize space across our warehouses, certain items need to be reduced. This will help make more room and improve efficiency in storage management.

Warehouse	Items to be Reduced
A	14
B	12
C	12
D	14

Increase

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, p.productScale,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock, p.productScale

),

InventoryAnalysis AS (
        SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item, ts.total_revenue,
        ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales,
        ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level_sales,
        ROUND(ts.total_revenue / 18, 0) AS avg_monthly_revenue, ts.productScale,
        CASE
                WHEN ts.total_ordered_item < 1 THEN 'Never ordered'
                WHEN ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked'
                WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN 'Understocked'
                ELSE 'Well-Stocked'
        END AS inventory_status
        FROM TotalSales ts

)

SELECT ia.productCode, ia.warehouseCode, ia.quantityInStock, ia.total_ordered_item, ia.total_revenue,

ia.avg_monthly_sales, ia.good_stock_level_sales, ia.avg_monthly_revenue, ia.inventory_status,

CASE
        WHEN ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND
        (ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE '%1:50%'
        OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%') THEN 'Reduce'
        WHEN ia.inventory_status = 'Understocked' THEN 'Increase'
        ELSE 'Maintain'

END AS action

FROM InventoryAnalysis ia

where ia.inventory_status = 'Understocked'

ORDER BY ia.warehouseCode
```

Across all warehouses, there are 9 products that should have their stock levels increased to meet demand and avoid potential stockouts.

productCode	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	good_stock_level_sales	avg_monthly_revenue	inventory_status	action
S700_3167	a	551	1047	76618.4	58	698	4257	Understocked	Increase
S32_1374	a	178	1014	89364.89	56	676	4965	Understocked	Increase
S50_4713	a	600	992	73670.64	55	661	4093	Understocked	Increase
S24_2000	a	15	1015	67193.49	56	677	3733	Understocked	Increase
S12_1099	b	68	933	161531.48	52	622	8974	Understocked	Increase
S18_2795	c	548	880	132275.98	49	587	7349	Understocked	Increase
S18_2248	c	540	832	45306.77	46	555	2517	Understocked	Increase
S32_4289	c	136	972	60493.33	54	648	3361	Understocked	Increase
S72_3212	d	414	958	47550.4	53	639	2642	Understocked	Increase

Maintain

```
WITH TotalSales AS (
        SELECT p.productCode, p.productName, p.warehouseCode, p.quantityInStock, p.productScale,
        COALESCE(SUM(od.quantityOrdered), 0) AS total_ordered_item,
        COALESCE(SUM(od.quantityOrdered * od.priceEach), 0) AS total_revenue
        FROM mintclassics.products p
        LEFT JOIN mintclassics.orderdetails od ON p.productCode = od.productCode
        LEFT JOIN mintclassics.orders o ON od.orderNumber = o.orderNumber AND o.status IN ('Shipped', 'Resolved')
        GROUP BY p.productCode, p.warehouseCode, p.quantityInStock, p.productScale

),

InventoryAnalysis AS (
        SELECT ts.productCode, ts.productName, ts.warehouseCode, ts.quantityInStock, ts.total_ordered_item, ts.total_revenue,
        ROUND(ts.total_ordered_item / 18, 0) AS avg_monthly_sales,
        ROUND((ts.total_ordered_item / 18) * 12, 0) AS good_stock_level_sales,
        ROUND(ts.total_revenue / 18, 0) AS avg_monthly_revenue, ts.productScale,
        CASE
                WHEN ts.total_ordered_item < 1 THEN 'Never ordered'
                WHEN ts.quantityInStock > (ts.total_ordered_item / 18) * 12 THEN 'Overstocked'
                WHEN ts.quantityInStock < (ts.total_ordered_item / 18) * 12 THEN 'Understocked'
                ELSE 'Well-Stocked'
        END AS inventory_status
        FROM TotalSales ts

)

SELECT ia.productCode, ia.warehouseCode, ia.quantityInStock, ia.total_ordered_item, ia.total_revenue,

ia.avg_monthly_sales, ia.good_stock_level_sales, ia.avg_monthly_revenue, ia.inventory_status,

CASE
        WHEN ia.inventory_status = 'Never ordered' OR ia.inventory_status = 'Overstocked' AND
        (ia.productScale LIKE '%1:700%' OR ia.productScale LIKE '%1:72%' OR ia.productScale LIKE '%1:50%'
        OR ia.productScale LIKE '%1:32%' OR ia.productScale LIKE '%1:24%') THEN 'Reduce'
        WHEN ia.inventory_status = 'Understocked' THEN 'Increase'
        ELSE 'Maintain'

END AS action

FROM InventoryAnalysis ia

HAVING action = 'Maintain'

ORDER BY ia.total_ordered_item desc

Limit 10;
```

productCode	warehouseCode	quantityInStock	total_ordered_item	total_revenue	avg_monthly_sales	good_stock_level_sales	avg_monthly_revenue	inventory_status	action
S18_3232	b	8347	1808	276839.98	100	1205	15380	Overstocked	Maintain
S18_1342	c	8693	1111	102563.52	62	741	5698	Overstocked	Maintain
S18_3856	c	2378	1076	102537.45	60	717	5697	Overstocked	Maintain
S18_4600	d	3128	1061	114232.79	59	707	6346	Overstocked	Maintain
S10_1678	a	7933	1057	90157.77	59	705	5009	Overstocked	Maintain
S12_4473	d	6125	1056	109946.21	59	704	6108	Overstocked	Maintain
S18_2319	d	8258	1053	117669.66	59	702	6537	Overstocked	Maintain
S24_3856	b	6600	1052	134240.71	58	701	7458	Overstocked	Maintain
S18_1662	a	5330	1040	144959.91	58	693	8053	Overstocked	Maintain
S18_2949	c	4189	1038	97193.88	58	692	5400	Overstocked	Maintain

Across all warehouses, 53 items have been identified as having optimal stock levels, indicating that no immediate action is required for these products. Of these, 10 items are highlighted here as examples. Maintaining these stock levels ensures continued efficiency and avoids unnecessary costs.

Conclusion

Key Findings

- **Inventory Distribution**:
  Identified overstocked and understocked items across all warehouses.

- **Sales Performance:**
  Analyzed sales data to determine top-performing and low-performing products.

- **Warehouse Efficiency:**
  Assessed the efficiency and space utilization of each warehouse.

- **Financial Insights:**
  Evaluated the financial performance of different product lines.

Recommendations

- **Warehouse Optimization:**
  Reduce or reallocate overstocked items to cut storage costs and optimize space.

- **Stock Replenishment:**
  Increase stock levels of understocked items to prevent stockouts and meet demand.

- **Warehouse Closure:**
  Close Warehouse C by relocating its inventory to other warehouses.
  For example, move cars from Warehouse C to Warehouse B, which also stocks cars, after reducing some items in Warehouse B & Warehouse C.

- **Sales Strategy:**
  Focus marketing efforts on low-performing products to boost their sales.

- **Continuous Monitoring:**
  Implement ongoing inventory and sales performance monitoring for sustained efficiency.